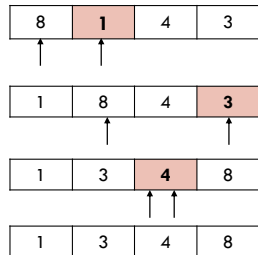


## selection sort

2

- Zoek de kleinste waarde in de lijst.
- Verwissel het met de eerste waarde in de lijst.
- Herhaal deze stappen met de rest van de lijst.



Complexiteit is  $O(n^2)$

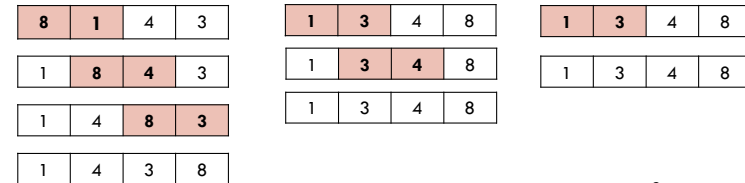


## bubble sort

3



- Vergelijk paren, wissel indien nodig.
- Overloop de lijst: laatste = grootste
- Herhaal deze stappen met kortere lijst.



Complexiteit is  $O(n^2)$

## Implementatie

4

- Vergelijken met generic type T :
  - ▣ `IComparer<T>` noodzakelijk
  - ▣ Gebruik default-comparer
- `Array.Sort(...)` gebruikt:
  - ▣ indien  $n < 16$  : Insertion sort
  - ▣  $n \gg$  Heap sort
  - ▣ Anders : Quick sort

## Opmerkingen

5

Sorteer Boek-objecten:

- Voeg .dll toe
- Unit-testen :
  - ▣ Elk boek-item maar 1 keer aanmaken

## BestandSorteerder<T>

6

Comparer bijhouden in instantievariabele:

```
IComparer<T> vergelijker;
```

□ Klasse aanmaken (voor Park / School):

```
public class ParkSorteer : IComparer<Park>
//bevat methode :
public int Compare(Park x, Park y){ .... }
```

□ comparer toekennen aan de variabele:

```
vergelijker = new ParkSorteer();
```


## BestandSorteerder<T>

7

De methodes bijhouden in instantievariabelen:

□ met delegate //buiten de klasse

```
public delegate T LeesType<T>(string lijn);
public delegate void SortType<T>(IList<T> l, IComparer<T> c);
```



□ Instantievariabelen in de klasse:

```
public LeesType<T> LeesMethode ;
public SortType<T> SorteerMethode;
```

## BestandSorteerder<T>

8

De methodes bijhouden in instantievariabelen:

□ met Func / Action (void-functie) – in de klasse

```
public Func<string, T> LeesMethode;
public Action<IList<T>, IComparer<T>> SorteerMethode
```




## BestandSorteerder<T>

9

□ Methode toekennen aan de variabele:

```
LeesMethode = SchoolLezer.LeesSchool;
```

```
//Signatuur:
School LeesSchool(string invoer)
```



```
SorteerMethode = SorteerBib<School>.BubbleSorteer
```

```
//Signatuur:
void BubbleSorteer(IList<T> objecten, IComparer<T> compare)
```